시험에 나오는 것만 공부한다! 시나공

이번에도 출제될 필수 코드 57문제

정보처리기사 필기

1. 다음 C언어 프로그램이 실행되었을 때, 실행 결과는?

```
#include <stdio.h>
#include <stdlib.h>
int main(int argc, char* argv[ ]) {
    char str1[20] = "KOREA";
    char str2[20] = "LOVE";
    char* p1 = NULL;
    char* p2 = NULL;
    p1 = str1;
    p2 = str2;
    str1[1] = p2[2];
    str2[3] = p1[4];
    strcat(str1, str2);
    printf("%c", *(p1 + 2));
    return 0;
}
```

① E
② V
③ R
④ O

[해설]
사용된 코드의 의미는 다음과 같습니다.

```
#include <stdio.h>
#include <stdlib.h>
int main(int argc, char* argv[]) {
❶   char str1[20] = "KOREA";
❷   char str2[20] = "LOVE";
❸   char* p1 = NULL;
❹   char* p2 = NULL;
❺   p1 = str1;
❻   p2 = str2;
❼   str1[1] = p2[2];
❽   str2[3] = p1[4];
❾   strcat(str1, str2);
❿   printf("%c", *(p1 + 2));
⓫   return 0;
}
```

❶ 20개의 요소를 갖는 문자형 배열 str1을 선언하고 "KOREA"로 초기화합니다.

	[0]	[1]	[2]	[3]	[4]	[5]	…	[19]
str1	'K'	'O'	'R'	'E'	'A'	\0	…	

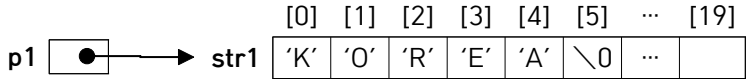
❷ 20개의 요소를 갖는 문자형 배열 str2를 선언하고 "LOVE"로 초기화합니다.

	[0]	[1]	[2]	[3]	[4]	…	[19]
str2	'L'	'O'	'V'	'E'	\0	…	

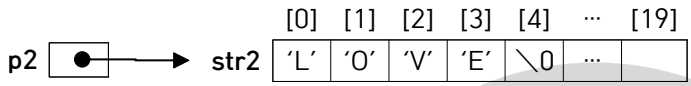
❸ 문자형 포인터 변수 p1을 선언하고 NULL로 초기화합니다.

❹ 문자형 포인터 변수 p2를 선언하고 NULL로 초기화합니다.

❺ p1에 str1 배열의 시작 주소를 저장합니다.



❻ p2에 str2 배열의 시작 주소를 저장합니다.



❼ p2는 str2를 가리키므로 str2[2]의 값인 'V'를 str1[1]에 저장합니다.

	[0]	[1]	[2]	[3]	[4]	[5]	…	[19]
str1	'K'	'V'	'R'	'E'	'A'	\0	…	

❽ p1은 str1을 가리키므로 str1[4]의 값인 'A'를 str2[3]에 저장합니다.

	[0]	[1]	[2]	[3]	[4]	…	[19]
str2	'L'	'O'	'V'	'A'	\0	…	

❾ str1의 문자열 뒤에 str2의 문자열을 이어붙입니다.

- **strcat(문자배열A, 문자배열B)** : A 배열에 저장된 문자열의 마지막에 이어서 B 배열에 저장된 문자열을 이어붙임

	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	…	[19]
str1	'K'	'V'	'R'	'E'	'A'	'L'	'O'	'V'	'A'	\0	…	

❿ p1+2가 가리키는 곳의 값을 문자로 출력합니다. p1은 str1 배열의 시작주소, 즉 str1[0]의 위치를 가리키므로, p1+2는 str1[0]의 다음 두 번째 요소인 'R'을 가리킵니다.

결과 R

⓫ main( ) 함수에서의 'return 0'은 프로그램의 종료를 의미합니다.

2. 다음 C언어 프로그램이 실행되었을 때, 실행 결과는?

```
#include <stdio.h>
#include <stdlib.h>
int main(int argc, char* argv[ ]) {
    int arr[2][3] = { 1,2,3,4,5,6 };
    int (*p)[3] = NULL;
    p = arr;
    printf("%d, ", *(p[0] + 1) + *(p[1] + 2));
    printf("%d", *(*(p + 1) + 0) + *(*(p + 1) + 1));
    return 0;
}
```

① 7, 5
② 8, 5
③ 8, 9
④ 7, 9

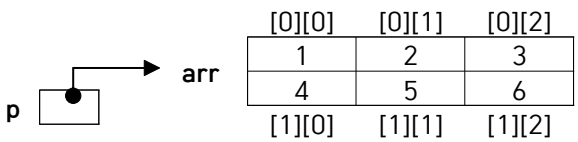
[해설]
사용된 코드의 의미는 다음과 같습니다.

```
#include   <stdio.h>
#include   <stdlib.h>
int main(int argc, char* argv[]) {
❶   int arr[2][3] = { 1,2,3,4,5,6 };
❷   int (*p)[3] = NULL;
❸   p = arr;
❹   printf("%d, ", *(p[0] + 1) + *(p[1] + 2));
❺   printf("%d", *(*(p + 1) + 0) + *(*(p + 1) + 1));
❻   return 0;
}
```

❶ 2행 3열의 요소를 갖는 정수형 2차원 배열 arr을 선언하고 초기화합니다.

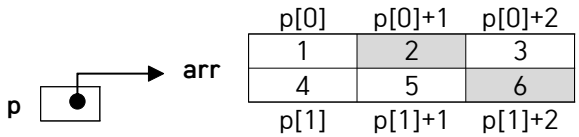
	[0][0]	[0][1]	[0][2]
arr	1	2	3
	4	5	6
	[1][0]	[1][1]	[1][2]

❷ 3개의 요소를 갖는 정수형 포인터 배열 p를 선언하고 NULL로 초기화합니다.
❸ p에 arr의 주소를 저장합니다.



❹ printf("%d, ", *(p[0] + 1) + *(p[1] + 2));
㉠ ㉡

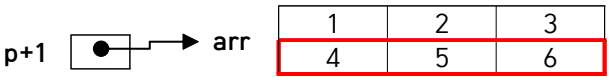
- ㉠ : p[0]은 arr 배열의 첫 번째 행의 시작 주소를 가리키므로 여기에 1을 더한다는 것은 1행의 두 번째 열의 값 2를 가리키는 것입니다.
- ㉡ : p[1]은 arr 배열의 두 번째 행의 시작 주소를 가리키므로 여기에 2를 더한다는 것은 2행의 세 번째 열의 값 6을 가리키는 것입니다.
- ㉠의 값 2와 ㉡의 값 6을 더한 값 8을 정수로 출력한 후 이어서 쉼표(,)와 공백 한 칸을 출력합니다.



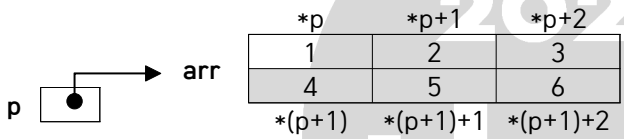
결과 8,

❺ printf("%d", *(*(p + 1) + 0) + *(*(p + 1) + 1));
㉠ ㉡

- 2차원 배열에서 배열명은 실제 값에 해당하는 요소가 아닌 첫 번째 행의 주소를 가리킵니다. 즉, p 또는 arr은 배열의 첫 번째 요소인 1을 가리키는 것이 아닌 첫 번째 행 전체를 가리키므로 만약 두 번째 행을 1차원 배열의 포인터처럼 사용하려면 ❹번에서와 같이 p[1]을 사용하거나 *(p+1)을 사용해야 합니다.



- ㉠ : *(p+1)은 arr 배열의 두 번째 행의 시작 주소를 가리키므로 여기에 0을 더한다는 것은 2행의 첫 번째 열의 값 4를 가리키는 것입니다.
- ㉡ : *(p+1)은 arr 배열의 두 번째 행의 시작 주소를 가리키므로 여기에 1을 더한다는 것은 2행의 두 번째 열의 값 5를 가리키는 것입니다.
- ㉠의 값 4와 ㉡의 값 5를 더한 값 9를 정수로 출력합니다.



결과 8, 9

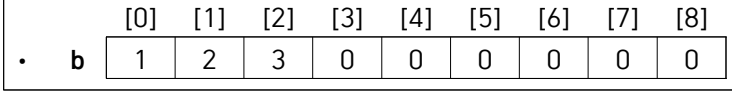
❻ main( ) 함수에서의 'return 0'은 프로그램의 종료를 의미합니다.

3. C 언어에서 배열 b[5]의 값은?

```
static int b[9]={1, 2, 3};
```

① 0
② 1
③ 2
④ 3

[해설]
- C 언어의 변수 선언 시 앞에 static을 붙이면 정적 변수로 선언됩니다.
- 정적 변수는 초기화를 생략하면 0으로 자동 초기화 됩니다.



4. 다음 C언어 프로그램이 실행되었을 때의 결과는?

```
#include <stdio.h>
int main(int argc, char *argv[ ]) {
    int a[2][2] = {{11, 22}, {44, 55}};
    int i, sum = 0;
    int *p;
    p = a[0];
    for(i = 1; i < 4; i++)
        sum += *(p + i);
    printf("%d", sum);
    return 0;
}
```

① 55
② 77
③ 121
④ 132

[해설]
사용된 코드의 의미는 다음과 같습니다.

```
#include <stdio.h>
int main(int argc, char *argv[ ]) {
❶   int a[2][2] = {{11, 22}, {44, 55}};
❷   int i, sum = 0;
❸   int *p;
❹   p = a[0];
❺   for(i = 1; i < 4; i++)
❻       sum += *(p + i);
❼   printf("%d", sum);
❽   return 0;
}
```

❶ 2행 2열의 요소를 갖는 정수형 2차원 배열 a를 선언하고 초기화합니다.

a 배열

	a[0][0]	a[0][1]
	11	22
	44	55
	a[1][0]	a[1][1]

❷ 정수형 변수 i, sum을 선언하고, sum을 0으로 초기화합니다.
❸ 정수형 포인터 변수 p를 선언합니다.
❹ p에 a배열의 a[0]의 주소를 저장합니다.
※ a[0]은 0행의 0번째 요소(a[0][0])의 주소입니다.
❺ 반복 변수 i가 1부터 1씩 증가하면서 4보다 작은 동안 ❻번을 반복 수행합니다.
❻ sum에 p+i가 가리키는 곳의 값을 더합니다.
• p는 a[0][0]을 가리키므로 숫자가 더해진 만큼 다음 값을 가리키게 됩니다. 즉, p+1은 a[0][1]을, p+2는 a[1][0]을, p+3은 a[1][1]을 가리킵니다.
• 반복문 실행에 따른 변수의 변화는 다음과 같습니다.

반복횟수	i	*(p+i)	sum
			0
1	1	22	22
2	2	44	66
3	3	55	121
반복실행 안됨	4		

❼ sum의 값 121을 정수로 출력합니다.

결과 121

❽ 프로그램을 종료합니다.

5. 다음 C언어 프로그램에서 밑줄 친 부분과 동일한 의미를 가지는 것은 어떤 것인가?

```
#include <stdio.h>
main( ) {
    int a, b;
    for (a = 0; a < 2; a++)
        for (b = 0; b < 2; b++)
            printf("%d", !a && !b);
}
```

① !a || !b

② !(a || b)

③ a && b

④ a || b

[해설]

• !a && !b는 불 대수로 변환하면 $\bar{a}\cdot\bar{b}$가 됩니다. $\bar{a}\cdot\bar{b}$는 드모르강 정리에 의해 $\overline{a+b}$이므로, 이것을 다시 조건식으로 변환하면 !(a || b)가 됩니다.

• 드모르강 법칙

- $\overline{A+B} = \bar{A}\cdot\bar{B}$
- $\overline{A\cdot B} = \bar{A}+\bar{B}$

• 불 대수와 드모르강 정리를 모르더라도 다음과 같이 a와 b에 들어갈 수 있는 값들을 대입하여 같은 결과를 내는 조건식을 찾을 수 있습니다.

• **!a && !b**

a	b	!a	!b	!a && !b
0	0	1	1	1
0	1	1	0	0
1	0	0	1	0
1	1	0	0	0

• **!(a || b)**

a	b	a || b	!(a || b)
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

6. 다음 C언어 프로그램이 실행되었을 때의 결과는?

```
#include <stdio.h>
int main(int argc, char *argv[ ]) {
  char a;
  a = 'A' + 1;
  printf("%d", a);
  return 0;
}
```

① 1
② 11
③ 66
④ 98

[해설]
사용된 코드의 의미는 다음과 같습니다.

```
#include <stdio.h>
int main(int argc, char *argv[ ]) {
❶ char a;
❷ a = 'A' + 1;
❸ printf("%d", a);
❹ return 0;
}
```

❶ 문자형 변수 a를 선언합니다.
❷ a에 문자 'A'와 숫자 1을 더한 값을 저장합니다.
※ 'A'라는 문자는 메모리에 저장될 때 문자로 저장되는 것이 아니라 해당 문자의 아스키 코드 값으로 저장됩니다. 즉, 'A'는 'A'에 해당하는 아스키 코드 값인 65가 저장되는 것이죠. 그러므로 a에는 'A'의 아스키 코드 값인 65에 1을 더한 값인 66이 저장됩니다.
❸ a의 값 66을 정수로 출력합니다.
※ a에 저장된 66은 "%d"로 출력하면 정수 66이, "%c"로 출력하면 'A'의 다음 문자인 'B'가 출력됩니다.
결과 66
❹ 프로그램을 종료합니다.

7. 다음 C 프로그램의 결과 값은?

```
main(void) {
  int i;
  int sum = 0;
  for(i = 1; i <= 10; i = i + 2)
       sum = sum + i;
  printf("%d", sum);
}
```

① 15
② 19
③ 25
④ 27

[해설]
사용된 코드의 의미는 다음과 같습니다.

```
main(void) {
❶ int i;
❷ int sum = 0;
❸ for(i = 1; i <= 10; i = i + 2)
❹       sum = sum + i;
❺ printf("%d", sum);
}
```

❶ 정수형 변수 i를 선언합니다.
❷ 정수형 변수 sum을 선언하고 0으로 초기화합니다.
❸ 반복 변수 i가 1에서 2씩 증가하면서 10보다 작거나 같은 동안 ❹번을 반복 수행합니다.
❹ sum에 i의 값을 누적합니다.
반복문 실행에 따른 변수들의 변화는 다음과 같습니다.

반복횟수	i	sum
		0
1	1	1
2	3	4
3	5	9
4	7	16
5	9	25
	11	

❺ sum의 값을 출력합니다.
결과 25

8. 다음 C언어 프로그램이 실행되었을 때의 결과는?

```
#include <stdio.h>
int main(int argc, char *argv[ ]) {
    int a = 4;
    int b = 7;
    int c = a | b;
    printf("%d", c);
    return 0;
}
```

① 3
② 4
③ 7
④ 10

[해설]
사용된 코드의 의미는 다음과 같습니다.

```
#include <stdio.h>
int main(int argc, char *argv[ ] {
❶   int a = 4;
❷   int b = 7;
❸   int c = a | b;
❹   printf("%d", c);
❺   return 0;
}
```

❶ 정수형 변수 a를 선언하고 4로 초기화합니다.
❷ 정수형 변수 b를 선언하고 7로 초기화합니다.
❸ 정수형 변수 c를 선언하고 a의 값 4와 b의 값 7을 |(비트 or)연산한 값으로 초기화합니다.
  • |(비트 or)는 두 비트 중 한 비트라도 1이면 1이 되는 비트 연산자입니다.
    4 = 0 0 0 0 0 1 0 0
    7 = 0 0 0 0 0 1 1 1
    | 0 0 0 0 0 1 1 1 (7)
  • c에는 7이 저장됩니다.
❹ c의 값을 정수로 출력합니다.
  결과 7
❺ 프로그램을 종료합니다.

9. 다음 C언어 프로그램이 실행되었을 때, 실행 결과는?

```
#include <stdio.h>
struct st {
    int a;
    int c[10];
};
int main(int argc, char* argv[ ]) {
    int i = 0;
    struct st ob1;
    struct st ob2;
    ob1.a = 0;
    ob2.a = 0;
    for (i = 0; i < 10; i++) {
        ob1.c[i] = i;
        ob2.c[i] = ob1.c[i] + i;
    }
    for (i = 0; i < 10; i = i + 2) {
        ob1.a = ob1.a + ob1.c[i];
        ob2.a = ob2.a + ob2.c[i];
    }
    printf("%d", ob1.a + ob2.a);
    return 0;
}
```

① 30
② 60
③ 80
④ 120

[해설]
사용된 코드의 의미는 다음과 같습니다.

```
#include <stdio.h>
struct st {             구조체 st를 정의합니다.
    int a;              정수형 변수 a를 선언합니다.
    int c[10];          10개의 요소를 갖는 정수형 배열 c를 선언합니다.
};
int main(int argc, char* argv[]) {
❶   int i = 0;
❷   struct st ob1;
❸   struct st ob2;
❹   ob1.a = 0;
❺   ob2.a = 0;
❻   for (i = 0; i < 10; i++) {
❼       ob1.c[i] = i;
❽       ob2.c[i] = ob1.c[i] + i;
    }
```

```
❾  for (i = 0; i < 10; i = i + 2) {
❿       ob1.a = ob1.a + ob1.c[i];
⓫       ob2.a = ob2.a + ob2.c[i];
    }
⓬  printf("%d", ob1.a + ob2.a);
⓭  return 0;
}
```

❶ 정수형 변수 i를 선언하고 0으로 초기화합니다.

❷ 구조체 st의 변수 ob1을 선언합니다.

	int a	int c[10]
ob1	ob1.a	ob1.c[0] ~ ob1.c[9]

❸ 구조체 st의 변수 ob2를 선언합니다.

	int a	int c[10]
ob2	ob2.a	ob2.c[0] ~ ob2.c[9]

❹ ob1.a에 0을 저장합니다.

❺ ob2.a에 0을 저장합니다.

❻ 반복 변수 i가 0부터 1씩 증가하면서 10보다 작은 동안 ❼, ❽번을 반복 수행합니다.

❼ ob1.c[i]에 i의 값을 저장합니다.

❽ ob2.c[i]에 ob1.c[i]와 i를 합한 값을 저장합니다.

반복문 실행에 따른 변수들의 변화는 다음과 같습니다.

i	ob1		ob2	
	a	c[i]	a	c[i]
0	0	0	0	0
1		1		2
2		2		4
3		3		6
4		4		8
5		5		10
6		6		12
7		7		14
8		8		16
9		9		18
10				

❾ 반복 변수 i가 0부터 2씩 증가하면서 10보다 작은 동안 ❿, ⓫번을 반복 수행합니다.

❿ ob1.a에 ob1.c[i]의 값을 누적시킵니다.

⓫ ob2.a에 ob2.c[i]의 값을 누적시킵니다.

반복문 실행에 따른 변수들의 변화는 다음과 같습니다.

i	ob1		ob2	
	a	c[i]	a	c[i]
0	0	0	0	0
2	2	2	4	4
4	6	4	12	8
6	12	6	24	12
8	20	8	40	16
10				

⓬ ob1.a와 ob2.a의 값을 합하여 정수로 출력합니다.

결과 60

⓭ main() 함수에서의 'return 0'은 프로그램의 종료를 의미합니다.

10. 다음 C언어 프로그램이 실행되었을 때의 결과는?

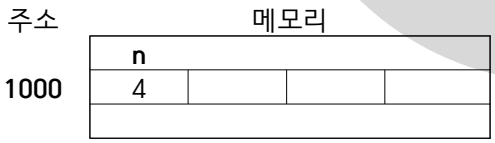
```
#include <stdio.h>
int main(void) {
   int n = 4;
   int* pt = NULL;
   pt = &n;
   printf("%d", &n + *pt - *&pt + n);
   return 0;
}
```

① 0
② 4
③ 8
④ 12

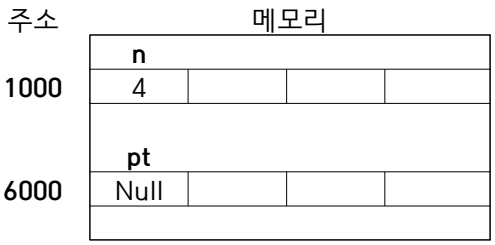
[해설]
사용된 코드의 의미는 다음과 같습니다.

```
#include <stdio.h>
int main(void) {
❶  int n = 4;
❷  int* pt = NULL;
❸  pt = &n;
❹  printf("%d", &n + *pt - *&pt + n);
❺  return 0;
}
```

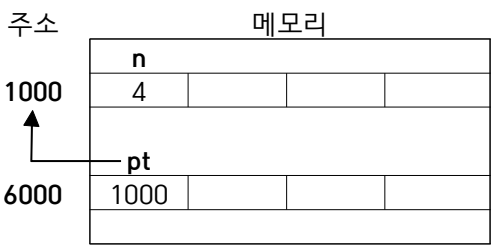
❶ 정수형 변수 n을 선언하고 4로 초기화합니다.
(다음 그림에서 지정된 주소는 임의로 정한 것이며, 이해를 돕기 위해 10진수로 표현했습니다.)



❷ 정수형 포인터 변수 pt에 Null 값을 저장합니다.



❸ pt에 n의 주소를 저장합니다.



❹ printf("%d", &n + *pt - *(&pt) + n);
ⓐ ⓑ ⓒ ⓓ ⓔ

주소 메모리

	n			
ⓐ 1000	4 ⓑ,ⓔ			
	pt			
ⓒ 6000	1000 ⓓ			

- ⓐ : n의 주소는 1000입니다.
- ⓑ : pt에 저장된 1000이 가리키는 곳의 값은 4입니다.
- ⓒ : pt의 주소는 6000입니다.
- ⓓ : 6000이 가리키는 곳의 값은 1000입니다.
- ⓔ : n의 값은 4입니다.
- ⓐ+ⓑ-ⓓ+ⓔ의 결과 8을 정수로 출력합니다.

결과 8

❺ 프로그램을 종료합니다.

11. C언어에서 정수 변수 a, b에 각각 1, 2가 저장되어 있을 때 다음 식의 연산 결과로 옳은 것은?

```
a < b + 2 && a << 1 <= b
```

① 0
② 1
③ 3
④ 5

[해설]
- 연산자의 우선순위는 높은 것부터 차례대로 단항, 산술, 시프트, 관계, 비트, 논리, 조건, 대입, 순서 연산자 순이며, 관계 연산자 중에서 <, <=, >=, >는 ==, !=보다 우선순위가 높습니다.
- 우선순위에 따라 문제의 식을 풀면 다음과 같습니다.

a < b + 2 && a << 1 <= b
❶ ❷ ❸ ❹ ❺

❶ b+2 : b의 값은 2이므로 결과는 4입니다.
❷ a<<1 : 왼쪽 시프트(<<)는 왼쪽으로 1비트 시프트 할 때마다 2배씩 증가하므로, a의 값 1을 왼쪽으로 1비트 시프트한 결과는 2입니다.
❸ a<❶ → a<4 : a의 값 1은 4보다 작으므로 결과는 1(참)입니다.
❹ ❷<=b → 2<=b : b의 값 2는 2와 같으므로 결과는 1(참)입니다.
❺ ❸&&❹ → 1&&1 : &&은 모두 참일 때만 참이므로 결과는 1(참)입니다.

12. a[0]의 주소값이 10일 경우 다음 C언어 프로그램이 실행되었을 때의 결과는? (단, int 형의 크기는 4Byte로 가정한다.)

```
#include <stdio.h>
int main(int argc, char* argv[ ]) {
   int a[ ] = { 14,22,30,38 };
   printf("%u, ", &a[2]);
   printf("%u", a);
   return 0;
}
```

① 14, 10
② 14, M
③ 18, 10
④ 18, M

[해설]
사용된 코드의 의미는 다음과 같습니다.

```
#include <stdio.h>
int main(int argc, char* argv[ ]) {
❶  int a[ ] = { 14,22,30,38 };
❷  printf("%u, ", &a[2]);
❸  printf("%u", a);
❹  return 0;
}
```

❶ 4개의 요소를 갖는 정수형 배열 a를 선언하고 초기화합니다.

❷ a[2]의 주소를 부호없는 정수형으로 출력하고, 이어서 쉼표(,)와 공백 한 칸을 출력합니다. a[0]의 주소가 10이고, 정수형(int)의 크기가 4Byte이므로 a[2]의 주소로 **18,** 이 출력됩니다.

주소	10	14	18	22	26
	[0]	[1]	[2]	[3]	
a	14	22	30	38	

결과 18,

❸ a를 부호없는 정수형으로 출력한다. 배열의 이름은 배열의 시작 주소를 의미하므로 10이 출력됩니다.

결과 18, 10

❹ main( ) 함수에서의 'return 0'은 프로그램의 종료를 의미합니다.

13. 다음 C언어 프로그램이 실행되었을 때, 실행 결과는?

```
#include <stdio.h>
int main(int arge, char* argv[ ]) {
    int n1 = 1, n2 = 2, n3 = 3;
    int r1, r2, r3;
    r1 = (n2 <= 2) || (n3 > 3);
    r2 = !n3;
    r3 = (n1 > 1) && (n2 < 3);
    printf("%d", r3 - r2 + r1);
    return 0;
}
```

① 0
② 1
③ 2
④ 3

[해설]
사용된 코드의 의미는 다음과 같습니다.

```
#include <stdio.h>
int main(int arge, char* argv[ ]) {
❶   int n1 = 1, n2 = 2, n3 = 3;
❷   int r1, r2, r3;
❸   r1 = (n2 <= 2) || (n3 > 3);
❹   r2 = !n3;
❺   r3 = (n1 > 1) && (n2 < 3);
❻   printf("%d", r3 - r2 + r1);
❼   return 0;
}
```

❶ 정수형 변수 n1, n2, n3를 선언하고, 각각 1, 2, 3으로 초기화합니다.
❷ 정수형 변수 r1, r2, r3를 선언합니다.
❸ r1 = (n2 <= 2) || (n3 > 3);
ⓐ ⓑ
ⓒ
- ⓐ : n2의 값 2는 2보다 작거나 같으므로 참(1)입니다.
- ⓑ : n3의 값 3은 3보다 크지 않으므로 거짓(0)입니다.
- ⓒ : ⓐ||ⓑ는 둘 중 하나라도 참이면 참이므로 참(1)입니다.
- r1에는 참(1)이 저장됩니다.

❹ n3의 값 3은 참이므로, r2에는 참의 부정인 거짓(0)이 저장됩니다.
※ 정수로 논리값(참, 거짓)을 판별하면 0은 거짓, 0이외의 수는 참으로 결정되어 저장됩니다.
❺ r3 = (n1 > 1) && (n2 < 3);
ⓐ ⓑ
ⓒ
- ⓐ : n1의 값 1은 1보다 크지 않으므로 거짓(0)입니다.
- ⓑ : n2의 값 2는 3보다 작으므로 참(1)입니다.
- ⓒ : ⓐ&&ⓑ는 모두 참일 때만 참이므로 거짓(0)입니다.

• r3에는 거짓(0)이 저장됩니다.

❻ r3-r2+r1의 결과(0-0+1) 1을 정수로 출력합니다.

결과 1

❼ main( ) 함수에서의 'return 0'은 프로그램의 종료를 의미합니다.

14. 다음 C언어 프로그램이 실행되었을 때의 결과는?

```
#include <stdio.h>
#include <string.h>
int main(void) {
    char str[50] = "nation";
    char *p2 = "alter";
    strcat(str, p2);
    printf("%s", str);
    return 0;
}
```

① nation

② nationalter

③ alter

④ alternation

[해설]

사용된 코드의 의미는 다음과 같습니다.

```
#include <stdio.h>
#include <string.h>
int main(void) {
❶   char str[50] = "nation";
❷   char *p2 = "alter";
❸   strcat(str, p2);
❹   printf("%s", str);
❺   return 0;
}
```

❶ 50개의 요소를 갖는 문자형 배열 str을 선언하고 "nation"으로 초기화합니다.

❷ 문자형 포인터 변수 p2를 선언하고, "alter"가 저장된 곳의 주소로 초기화합니다.

❸ str이 가리키는 문자열에 p2가 가리키는 문자열을 붙입니다.

• **strcat(문자열A, 문자열B)** : 문자열A의 뒤에 문자열B를 연결하여 붙이는 함수

❹ str을 문자열로 출력합니다.

결과 nationalter

❺ main( ) 함수에서의 'return 0'은 프로그램의 종료를 의미합니다.

15. 다음 C언어 프로그램이 실행되었을 때, 실행 결과는?

```
#include <stdio.h>
#include <stdlib.h>
int main(int argc, char* argv[ ]) {
    int i = 0;
    while (1) {
        if (i == 4) {
            break;
        }
        ++i;
    }
    printf("i = %d", i);
}
```

① i = 0
② i = 1
③ i = 3
④ i = 4

[해설]
사용된 코드의 의미는 다음과 같습니다.

```
#include <stdio.h>
#include <stdlib.h>
int main(int argc, char* argv[ ]) {
❶ int i = 0;
❷ while (1) {
❸       if (i == 4) {
❹           break;
        }
❺       ++i;
    }
❻   printf("i = %d", i);
}
```

❶ 정수형 변수 i를 선언하고 0으로 초기화합니다.
❷ 조건이 참(1)이므로 ❸~❺번을 무한 반복합니다.
❸ i의 값이 4이면 ❹번으로 이동하고, 아니면 ❺번으로 이동합니다.
❹ 반복문을 탈출하여 ❻번으로 이동합니다.
❺ 'i = i + 1;'과 동일합니다. i의 값을 1 증가시킵니다.
반복문 실행에 따른 변수의 변화는 다음과 같습니다.

반복 횟수	i
	0
1	1
2	2
3	3
4	4
5	

❻ i = 을 출력한 후 이어서 i의 값을 정수로 출력합니다.
결과 i = 4

16. 다음 C언어 프로그램 실행 후, 'c'를 입력하였을 때 출력 결과는?

```
#include <stdio.h>
main( ) {
  char ch;
  scanf("%c", &ch);
  switch (ch) {
  case 'a':
     printf("one ");
  case 'b':
     printf("two ");
  case 'c':
     printf("three ");
     break;
  case 'd':
     printf("four ");
     break;
  }
}
```

① one
② one two
③ three
④ one two three four

[해설]
사용된 코드의 의미는 다음과 같습니다.

```
#include <stdio.h>
main( ) {
❶ char ch;
❷ scanf("%c", &ch);
❸ switch (ch) {
   case 'a':
      printf("one ");
   case 'b':
      printf("two ");
❹ case 'c':
❺     printf("three ");
❻     break;
   case 'd':
      printf("four ");
      break;
   } ❼
}
```

❶ 문자형 변수 ch를 선언합니다.
❷ 문자를 입력받아 ch에 저장합니다. 문제에서 'c'를 입력한다고 하였으므로 ch에는 'c'가 저장됩니다.

❸ ch의 값 'c'에 해당하는 case를 찾아갑니다. ❹번으로 이동합니다.
❹ case 'c'의 시작점입니다.
❺ 화면에 three와 공백 한 칸을 출력합니다.
결과 three
❻ switch문을 벗어나 ❼번으로 이동합니다.
❼ main( ) 함수가 끝났으므로 프로그램을 종료합니다.

17. 다음 자바 프로그램의 조건문을 삼항 조건 연산자로 옳게 표현한 것은?

```
int i = 7, j = 9;
int k;
if (i > j)
  k = i - j;
else
  k = i + j;
```

① **int i = 7, j = 9;**
int k;
k = (i>j) ? (i - j) : (i + j);
② int i = 7, j = 9;
int k;
k = (i<j) ? (i - j) : (i + j);
③ int i = 7, j = 9;
int k;
k = (i>j) ? (i + j) : (i - j);
④ int i = 7, j = 9;
int k;
k = (i<j) ? (i + j) : (i - j);

[해설]
삼항 조건 연산자의 형식은 '조건 ? 참일 때 수식 : 거짓일 때 수식;'입니다.

```
❶ int i = 7, j = 9;
❷ int k;
❸ k = (i>j) ? (i - j) : (i + j);
```

❶ 정수형 변수 i와 j를 선언하고, 각각 7과 9로 초기화합니다.
❷ 정수형 변수 k를 선언합니다.
❸ i의 값이 j의 값보다 크면 k에 i-j의 값을 저장하고, 작거나 같으면 k에 i+j의 값을 저장합니다.

18. 다음 JAVA 프로그램이 실행되었을 때의 결과를 쓰시오.

```
public class ovr {
  public static void main(String[ ] args) {
    int arr[ ];
    int i = 0;
    arr = new int[10];
    arr[0] = 0;
    arr[1] = 1;
    while(i < 8) {
      arr[i + 2] = arr[i + 1] + arr[i];
      i++;
    }
    System.out.println(arr[9]);
  }
}
```

① 13
② 21
③ 34
④ 55

[해설]
사용된 코드의 의미는 다음과 같습니다

```
public class ovr {
  public static void main(String[ ] args) {
❶   int arr[ ];
❷   int i = 0;
❸   arr = new int[10];
❹   arr[0] = 0;
❺   arr[1] = 1;
❻   while(i < 8) {
❼     arr[i + 2] = arr[i + 1] + arr[i];
❽     i++;
    }
❾   System.out.println(arr[9]);
  }
}
```

❶ 정수형 배열 arr을 선언합니다.
❷ 정수형 변수 i를 선언하고 0으로 초기화합니다.
❸ arr에 10개의 요소를 할당합니다.
❹ arr[0]에 0을 저장합니다.
❺ arr[1]에 1을 저장합니다.
❻ i가 8보다 작은 동안 ❼, ❽번을 반복 수행합니다.
❼ arr[i+2]에 arr[i+1]과 arr[i]를 더한 값을 저장합니다.
❽ 'i = i + 1;'과 동일합니다. i에 1씩 저장시킵니다.

반복문을 수행한 결과는 다음과 같습니다.

반복 횟수	i	arr [0][1][2][3][4][5][6][7][8][9]
	0	0 1
1회	1	0 1 1
2회	2	0 1 1 2
3회	3	0 1 1 2 3
4회	4	0 1 1 2 3 5
5회	5	0 1 1 2 3 5 8
6회	6	0 1 1 2 3 5 8 13
7회	7	0 1 1 2 3 5 8 13 21
8회	8	0 1 1 2 3 5 8 13 21 34

❾ arr[9]의 값을 출력합니다.

결과 34

19. 다음 JAVA 코드 출력문의 결과는?

```
..생략..
System.out.println("5 + 2 = " + 3 + 4);
System.out.println("5 + 2 = " + (3 + 4));
..생략..
```

① 5 + 2 = 34
   5 + 2 = 34

② 5 + 2 + 3 + 4
   5 + 2 = 7

③ 7 = 7
   7 + 7

④ 5 + 2 = 34
   5 + 2 = 7

[해설]

Java의 print( ) 또는 println( )에서 **숫자 + 숫자**는 연산 결과를 숫자로 출력하고, **문자 + 숫자**는 두 값을 이어서 결과를 문자로 출력합니다.

```
❶ System.out.println("5 + 2 = " + 3 + 4);
❷ System.out.println("5 + 2 = " + (3 + 4));
```

❶ (("5 + 2 = " + 3) + 4)의 순서로 수행되며, ("5 + 2 =" + 3)는 **문자+숫자**이므로 "5 + 2 = 3"이라는 문자를, ("5 + 2 = 3" + 4) 또한 **문자+숫자**이므로 **5 + 2 = 34**라는 결과를 출력합니다.

❷ ("5 + 2 =" + (3 + 4))의 순서로 수행되며, 3 + 4는 **숫자+숫자**이므로 7이 되고, ("5 + 2 = " + 7)은 **문자+숫자**이므로 **5 + 2 = 7**이라는 결과를 출력합니다.

20. 다음 JAVA 프로그램이 실행되었을 때, 실행 결과는?

```
public class Ape {
    static void rs(char a[ ]) {
        for(int i = 0; i < a.length; i++)
            if(a[i] == 'B')
                a[i] = 'C';
            else if(i == a.length - 1)
                a[i] = a[i-1];
            else a[i] = a[i+1];
    }
    static void pca(char a[ ]) {
        for(int i = 0; i < a.length; i++)
            System.out.print(a[i]);
            System.out.println( );
    }
    public static void main(String[ ] args) {
        char c[ ] = {'A','B','D','D','A','B','C'};
        rs(c);
        pca(c);
    }
}
```

① BCDABCA
② BCDABCC
③ CDDACCC
④ CDDACCA

[해설]
사용된 코드의 의미는 다음과 같습니다.

```
public class Ape {
❸   static void rs(char a[]) {
❹       for(int i = 0; i < a.length; i++)
❺           if(a[i] == 'B')
❻               a[i] = 'C';
❼           else if(i == a.length - 1)
❽               a[i] = a[i-1];
❾           else a[i] = a[i+1];
    }
⓫   static void pca(char a[]) {
⓬       for(int i = 0; i < a.length; i ++)
⓭           System.out.print(a[i]);
⓮       System.out.println();
    }
    public static void main(String[] args) {
❶       char c[] = {'A','B','D','D','A','B','C'};
❷       rs(c);
⓾       pca(c);
```

```
    } ⓯
}
```

모든 Java 프로그램의 실행은 반드시 main( ) 메소드에서 시작합니다.

❶ 7개의 요소를 갖는 문자형 배열 c를 선언하고 초기화합니다.

	[0]	[1]	[2]	[3]	[4]	[5]	[6]
c	'A'	'B'	'D'	'D'	'A'	'B'	'C'

❷ 배열 c의 시작 위치를 인수로 rs( ) 메소드를 호출합니다.

❸ 메소드 rs( )의 시작점입니다. ❷번에서 전달받은 배열 c의 시작 위치를 배열 a가 받습니다.

a → c

[0]	[1]	[2]	[3]	[4]	[5]	[6]
'A'	'B'	'D'	'D'	'A'	'B'	'C'

❹ 반복 변수 i가 0에서 시작하여 1씩 증가하면서 배열 a의 길이인 7보다 작은 동안 ❺~❾번을 반복 수행합니다.

• **length** : 배열 요소의 개수를 저장하고 있는 속성

❺ a[i]가 'B'이면 ❻번으로 이동하고, 아니면 ❼번으로 이동합니다.

❻ a[i]에 'C'를 저장합니다.

❼ i의 값이 6(7-1)이면 ❽번으로 이동하고, 아니면 ❾번으로 이동합니다.

❽ a[i]에 a[i-1]의 값을 저장합니다.

❾ a[i]에 a[i+1]의 값을 저장합니다.

반복문 실행에 따른 변수의 변화는 다음과 같습니다.

i	a [0] [1] [2] [3] [4] [5] [6]
0	~~'A'~~ 'B' 'D' 'D' 'A' 'B' 'C' 'B'
1	'B' ~~'B'~~ 'D' 'D' 'A' 'B' 'C' 'C'
2	'B' 'C' ~~'D'~~ 'D' 'A' 'B' 'C' 'D'
3	'B' 'C' 'D' ~~'D'~~ 'A' 'B' 'C' 'A'
4	'B' 'C' 'D' 'A' ~~'A'~~ 'B' 'C' 'B'
5	'B' 'C' 'D' 'A' 'B' ~~'B'~~ 'C' 'C'
6	'B' 'C' 'D' 'A' 'B' 'C' ~~'C'~~ 'C'
7	

• 메소드가 종료되면 rs( ) 메소드를 호출했던 ❷번의 다음 줄인 ❿번으로 이동합니다.

❿ 배열 c의 시작 위치를 인수로 pca( ) 메소드를 호출합니다.

⓫ 메소드 pca( )의 시작점입니다. ❿번에서 전달받은 배열 c의 시작 위치를 새로운 배열 a가 받습니다.

a → c

[0]	[1]	[2]	[3]	[4]	[5]	[6]
'B'	'C'	'D'	'A'	'B'	'C'	'C'

⓬ 반복 변수 i가 0에서 시작하여 배열 a의 길이인 7보다 작은 동안 ⓭번을 반복 수행합니다.

⓭ a[i]의 값을 출력합니다.

결과 BCDABCC

⓮ 커서를 다음 줄의 처음으로 옮깁니다. 메소드가 종료되었으므로 pca( ) 메소드를 호출했던 ❿번의 다음 줄인 ⓯번으로 이동합니다.

⓯ 프로그램을 종료합니다.

21. 다음 JAVA 프로그램이 실행되었을 때의 결과는?

```
public class ovr {
  public static void main(String[ ] args) {
    int a = 1, b = 2, c = 3, d = 4;
    int mx, mn;
    mx = a < b ? b : a;
    if (mx == 1) {
        mn = a > mx ? b : a;
    }
    else {
        mn = b < mx ? d : c;
    }
    System.out.println(mn);
  }
}
```

① 1
② 2
③ 3
④ 4

[해설]
사용된 코드의 의미는 다음과 같습니다.

```
public class ovr {
  public static void main(String[ ] args) {
❶   int a = 1, b = 2, c = 3, d = 4;
❷   int mx, mn;
❸   mx = a < b ? b : a;
❹   if (mx == 1) {
❺       mn = a > mx ? b : a;
    }
    else {
❻       mn = b < mx ? d : c;
    }
❼   System.out.println(mn);
  }
}
```

❶ 정수형 변수 a, b, c, d를 선언하고, 각각 1, 2, 3, 4로 초기화합니다.
❷ 정수형 변수 mx, mn을 선언합니다.
❸ a가 b보다 작으면 mx에 b의 값을 저장하고, 아니면 a의 값을 저장합니다. a의 값 1은 b의 값 2보다 작으므로 mx에는 b의 값 2가 저장됩니다. (mx = 2)
❹ mx가 1이면 ❺번으로 이동하고, 아니면 ❻번으로 이동합니다. mx의 값은 2이므로 ❻번으로 이동합니다.
❻ b가 mx보다 작으면 mn에 d의 값을 저장하고, 아니면 c의 값을 저장합니다. b의 값 2는 mx의 값 2보다 작지 않으므로 mn에는 c의 값 3이 저장됩니다. (mn = 3)
❼ mn의 값 3을 출력하고 커서를 다음 줄의 처음으로 옮깁니다.

결과 3

22. 다음 JAVA 프로그램이 실행되었을 때, 실행 결과는?

```
public class Rarr {
  static int[ ] marr( ) {
    int temp[ ] = new int[4];
    for (int i = 0; i < temp.length; i++)
      temp[i] = i;
    return temp;
  }
public static void main(String[ ] args) {
  int iarr[ ];
  iarr = marr( );
  for (int i = 0; i < iarr.length; i++)
    System.out.print(iarr[i] + " ");
  }
}
```

① 1 2 3 4
② 0 1 2 3
③ 1 2 3
④ 0 1 2

[해설]
사용된 코드의 의미는 다음과 같습니다.

```
public class Rarr {
❸ static int[ ] marr( ) {
❹   int temp[ ] = new int[4];
❺   for (int i = 0; i < temp.length; i++)
❻     temp[i] = i;
❼   return temp;
  }
  public static void main(String[ ] args) {
❶   int iarr[ ];
❷❽ iarr = marr( );
❾   for (int i = 0; i < iarr.length; i++)
❿     System.out.print(iarr[i] + " ");
  }
}
```

모든 Java 프로그램은 반드시 main( ) 메소드에서 시작합니다.
❶ 정수형 배열 iarr을 선언합니다.
❷ marr( ) 메소드를 호출한 후 그 결과를 iarr에 저장합니다.
❸ 정수형 배열을 반환하는 marr( ) 메소드의 시작점입니다.
❹ 4개의 요소를 갖는 정수형 배열 temp를 선언합니다.
❺ 반복 변수 i가 0부터 1씩 증가하면서 temp의 길이인 4보다 작은 동안 ❻번을 반복 수합니다.
  • **length** : 배열 요소의 개수가 저장되어 있는 속성임. temp 배열은 4개의 요소를 가지므로 temp.length는 4를 가지고 있음
❻ temp[i]에 i의 값을 저장합니다.

반복문 실행에 따른 변수들의 변화는 다음과 같습니다.

i	temp[] [0] [1] [2] [3]
	∅ ∅ ∅ ∅
0	0
1	1
2	2
3	3
4	

❼ temp를 반환합니다. 인수나 반환값으로 배열의 이름을 지정하면 배열의 시작 주소가 전달됩니다. 즉 temp 배열의 시작 주소가 반환됩니다.

❽ ❼번에서 전달받은 정수형 배열 temp의 시작 주소를 iarr이 받습니다.

	[0]	[1]	[2]	[3]
iarr	0	1	2	3

❾ 반복 변수 i가 0부터 1씩 증가하면서 iarr의 길이인 4보다 작은 동안 ❿번을 반복 수행합니다.

❿ iarr[i]의 값을 출력한 후 공백 한 칸을 띄웁니다.

반복문 실행에 따른 출력 결과는 다음과 같습니다.

i	결과
0	0
1	0 1
2	0 1 2
3	0 1 2 3
4	

결과 0 1 2 3

23. 다음 JAVA 프로그램이 실행되었을 때의 결과는?

```
public class array1 {
  public static void main(String[ ] args) {
    int cnt = 0;
    do {
      cnt++;
    } while (cnt < 0);
    if(cnt== 1)
      cnt++;
    else
      cnt = cnt + 3;
    System.out.printf("%d", cnt);
  }
}
```

① 2
② 3
③ 4
④ 5

[해설]
사용된 코드의 의미는 다음과 같습니다.

```
public class array1 {
  public static void main(String[ ] args) {
❶   int cnt = 0;
❷   do {
❸       cnt++;
❹   } while (cnt < 0);
❺   if(cnt==1)
❻       cnt++;
    else
        cnt = cnt + 3;
❼   System.out.print("%d", cnt);
  }
}
```

❶ 정수형 변수 cnt를 선언하고 0으로 초기화합니다. (cnt=0)
❷ do~while 반복문의 시작점입니다. ❸번을 반복 수행합니다.
❸ 'cnt = cnt + 1;'과 동일합니다. cnt의 값을 1씩 누적시킵니다. (cnt=1)
❹ cnt가 0보다 작은 동안 ❸번을 반복 수행합니다.
do~while문은 조건이 거짓이라도 한 번은 실행하므로, cnt가 1이 된 후 do~while문을 빠져나옵니다.
❺ cnt가 1이면 ❻번을 수행하고, 아니면 else의 다음 문장을 수행합니다. cnt가 1이므로 ❻번으로 이동합니다.
❻ 'cnt = cnt + 1;'과 동일합니다. cnt의 값 1에 1을 누적시킵니다. (cnt=2)
❼ cnt의 값 2를 정수로 출력합니다.
결과 2

24. 다음 JAVA 프로그램이 실행되었을 때의 결과는?

```
public class Operator {
  public static void main(String[ ] args) {
     int x = 5, y = 0, z = 0;
     y = x++;
     z = --x;
     System.out.print(x + ", " + y + ", " + z);
  }
}
```

① 5, 5, 5
② 5, 6, 5
③ 6, 5, 5
④ 5, 6, 4

[해설]
사용된 코드의 의미는 다음과 같습니다.

```
public class Operator {
   public static void main(String[ ] args) {
❶     int x = 5, y = 0, z = 0;
❷     y = x++;
❸     z = --x;
❹     System.out.print(x + ", " + y + ", " + z);
   }
}
```

❶ 정수형 변수 x, y, z를 선언하고, 각각 5, 0, 0으로 초기화합니다. (x=5, y=0, z=0)
❷ x는 후치 증가 연산자이므로, x의 값 5를 y에 저장한 후 x의 값을 1 증가시킵니다. (x=6, y=5, z=0)
❸ x는 전치 감소 연산자이므로, x의 값을 1 감소시킨 후 x의 값 5를 z에 저장합니다. (x=5, y=5, z=5)
❹ x, y, z의 값을 ", "으로 구분하여 출력합니다.
결과 5, 5, 5

25. 다음 자바 코드를 실행한 결과는?

```
int x = 1, y = 6;
while (y--) {
    x++;
}
System.out.println("x = " x + "y = " y);
```

① x = 7 y = 0

② x = 6 y = -1

③ x = 7 y = -1

④ Unresolved compilation problem 오류 발생

[해설]

while문의 조건식과 println( ) 메소드의 인수 형식이 잘못되어서 오류가 발생하는 코드입니다. 올바르게 출력하려면, 다음과 같이 수정되어야 합니다.

```
❶ int x = 1, y = 6;
❷ while (y-- > 0) {
❸     x++;
  }
❹ System.out.println("x = " + x + " y = " + y);
```

❶ 정수형 변수 x, y를 선언하고 각각 1과 6으로 초기화합니다.

❷ y가 0보다 큰 동안 ❸번을 반복 수행합니다. y는 후치 감소 연산자이므로 조건식이 판별된 후 1 감소합니다.

❸ 'x = x + 1'과 동일합니다. x에 1씩 누적시킵니다.

❹ **x=**을 출력한 후 x의 값을, **y=**을 출력한 후 y의 값을 출력하고 커서를 다음 줄의 처음으로 이동시킵니다.

결과 x = 7 y = -1

26. 다음 파이썬으로 구현된 프로그램의 실행 결과로 옳은 것은?

```
>>> a = [0,10,20,30,40,50,60,70,80,90]
>>> a[:7:2]
```

① [20, 60]

② [60, 20]

③ [0, 20, 40, 60]

④ [10, 30, 50, 70]

[해설]

a[:7:2]는 배열 a의 0부터 6번째 위치까지 2씩 증가하면서 해당 위치의 요소를 출력하라는 의미입니다.

27. 다음 파이썬(Python) 프로그램이 실행되었을 때의 결과는?

```
def cs(n):
  s = 0
  for num in range(n+1):
     s += num
  return s
print(cs(11))
```

① 45
② 55
③ 66
④ 78

[해설]
사용된 코드의 의미는 다음과 같습니다.

```
❷   def cs(n):
❸       s = 0
❹       for num in range(n+1):
❺           s += num
❻       return s
❶❼ print(cs(11))
```

모든 Python 프로그램은 반드시 클래스 정의부가 종료된 이후의 코드에서 시작합니다.
❶ 11을 인수로 cs 메소드를 호출한 결과를 출력합니다.
❷ 메소드 cs의 시작점입니다. ❶번에서 전달받은 11을 n이 받습니다.
❸ s에 0을 저장합니다.
❹ n이 11이므로 range(12)가 되어 0에서 11을 순서대로 num에 저장하며 ❺번을 반복 수행합니다.
❺ s에 num의 값을 누적시킵니다.
반복문을 수행한 결과는 다음과 같습니다.

n	num	s
11	0	0
	1	1
	2	3
	3	6
	4	10
	5	15
	6	21
	7	28
	8	36
	9	45
	10	55
	11	66

❻ s의 값 66을 갖고 메소드를 호출했던 ❼번으로 이동합니다.
❼ ❻번에서 반환받은 값 66을 출력합니다.
결과 66

28. 다음은 파이썬으로 만들어진 반복문 코드이다. 이 코드의 결과는?

```
>> while(True) :
    print('A')
    print('B')
    print('C')
    continue
    print('D')
```

① A, B, C 출력이 반복된다.
② A, B, C
③ A, B, C, D 출력이 반복된다.
④ A, B, C, D 까지만 출력된다

[해설]
while(True)는 조건이 항상 참이므로 블록 내의 코드들을 무한 반복시키며, continue는 이후 코드를 수행하지 않고 반복문의 처음으로 돌아가는 예약어입니다. 따라서 화면에는 **A, B, C**만 반복하여 출력됩니다.

29. 다음은 사용자로부터 입력받은 문자열에서 처음과 끝의 3글자를 추출한 후 합쳐서 출력하는 파이썬 코드이다. ㉠에 들어갈 내용은?

```
String = input("7문자 이상 문자열을 입력하시오 :")
m = (  ㉠  )
print(m)
```

① string[1:3]+string[-3:]
② string[:3]+string[-3:-1]
③ **string[0:3]+string[-3:]**
④ string[0:]+string[:-1]

[해설]
문제에 제시된 보기들은 '객체명[초기위치:최종위치]'으로 기본 형식에서 '증가값'이 생략된 경우입니다. '증가값'이 생략된 경우에는 '초기위치'부터 '최종위치-1'까지 1씩 증가하면서 요소들을 가져옵니다. 각 보기의 의미는 다음과 같습니다.
① **string[1:3]+string[-3:]** : 1, 2번째 위치의 2글자와 -3, -2, -1번째 위치의 3글자를 가져옵니다.
② **string[:3]+string[-3:-1]** : 0, 1, 2번째 위치의 3글자와 -3, -2번째 위치의 2글자를 가져옵니다.
③ **string[0:3]+string[-3:]** : 0, 1, 2번째 위치의 3글자와 -3, -2, -1번째 위치의 3글자를 가져옵니다.
④ **string[0:]+string[:-1]** : 0부터 마지막 위치까지의 모든 글자와 첫 위치부터 -1까지의 모든 글자를 가져옵니다.
※ 문자열의 위치는 0부터 시작하며, -1위치는 문자열의 마지막 위치를 가리킵니다.
※ '초기위치', '최종위치'의 0은 생략이 가능합니다.

30. 다음 Python 프로그램이 실행되었을 때, 실행 결과는?

```
a = [“대”, “한”, “민”, “국”]
for i in a:
  print(i)
```

① 대한민국

② **대**
한
민
국

③ 대

④ 대대대대

[해설]

사용된 코드의 의미는 다음과 같습니다.

```
❶ a = [“대”, “한”, “민”, “국”]
❷ for i in a:
❸       print(i)
```

❶ 4개의 요소를 갖는 리스트 a를 선언하고 초기화합니다.

	[0]	[1]	[2]	[3]
a	“대”	“한”	“민”	“국”

❷ 반복 변수 i에 a의 각 요소들을 순서대로 저장하며 ❸번 문장을 반복 수행합니다.

❸ i의 값을 출력하고 커서를 다음 줄의 처음으로 옮깁니다.

※ 반복문 실행에 따른 변수의 변화는 다음과 같습니다.

반복 횟수	i	출력
1	“대”	**대**
2	“한”	대 **한**
3	“민”	대 한 **민**
4	“국”	대 한 민 **국**

31. 다음 파이썬(Python) 프로그램이 실행되었을 때의 결과는?

```
class FourCal:
  def setdata(self, fir, sec):
      self.fir = fir
      self.sec = sec
  def add(self):
      result = self.fir + self.sec
      return result
a = FourCal( )
a.setdata(4, 2)
print(a.add( ))
```

① 0
② 2
③ 4
④ 6

[해설]
사용된 코드의 의미는 다음과 같습니다.

```
ⓐ      class FourCal:
ⓑ❸        def setdata(self, fir, sec):
  ❹          self.fir = fir
  ❺          self.sec = sec
ⓒ❼        def add(self):
  ❽          result = self.fir + self.sec
  ❾          return result
  ❶    a = FourCal( )
  ❷    a.setdata(4, 2)
  ❻❿   print(a.add( ))
```

ⓐ 클래스 FourCal을 정의합니다.
ⓑ 2개의 인수를 받는 메소드 setdata( )를 정의합니다.
ⓒ 메소드 add( )를 정의합니다.
※ 모든 Python 프로그램은 반드시 클래스 정의부가 종료된 이후의 코드에서 시작합니다.
❶ FourCal 클래스의 객체 변수 a를 선언합니다.
❷ 4와 2를 인수로 a 객체의 setdata 메소드를 호출합니다.
❸ setdata 메소드의 시작점입니다. ❷번에서 전달받은 4와 2를 fir와 sec가 받습니다.
❹ a 객체에 변수 fir를 선언하고, fir의 값 4로 초기화합니다.
  • **self** : 클래스에 속한 메소드에 반드시 포함되어야 하는 예약어로, 메소드에서 자기 클래스에 속한 변수에 접근할 때 사용됨
❺ a 객체에 변수 sec를 선언하고, sec의 값 2로 초기화한다. 메소드가 종료되었으므로 메소드를 호출했던 ❷번의 다음 줄인 ❻번으로 이동합니다.
❻ a 객체의 add 메소드를 호출하고 반환받은 값을 출력합니다.
❼ add 메소드의 시작점입니다.
❽ 변수 result를 선언하고, a 객체의 변수 fir와 sec를 더한 값 6(4+2)으로 초기화합니다.
❾ result의 값 6을 메소드를 호출했던 곳으로 반환합니다.
❿ ❾번에서 반환받은 값 6을 출력합니다.
결과 6

32. 다음 Python 프로그램이 실행되었을 때, 실행 결과는?

```
a = 100
list_data = ['a','b','c']
dict_data = {'a':90, 'b':95}
print(list_data[0])
print(dict_data['a'])
```

①
```
a
90
```

②
```
100
90
```

③
```
100
100
```

④
```
a
a
```

[해설]

사용된 코드의 의미는 다음과 같습니다.

```
❶ a = 100
❷ list_data = ['a','b','c']
❸ dict_data = {'a':90, 'b':95}
❹ print(list_data[0])
❺ print(dict_data['a'])
```

❶ a에 100을 저장합니다.

❷ 3개의 요소를 갖는 리스트 list_data를 선언하고 초기화합니다.

	[0]	[1]	[2]
list_data	'a'	'b'	'c'

❸ 2개의 요소를 갖는 딕셔너리 dict_data를 선언하고 초기화합니다.

	['a']	['b']
dict_data	90	95

❹ list_data[0]의 값 **a**를 출력한 후 커서를 다음 줄의 처음으로 옮깁니다.

결과
```
a
```

❺ dict_data['a']의 값 **90**을 출력하고 커서를 다음 줄의 처음으로 옮깁니다.

결과
```
a
90
```

33. 다음 Python 프로그램의 실행 결과가 [실행 결과]와 같을 때, 빈칸에 적합한 것은?

```
x = 20
if x == 10:
   print('10')
(     ) x == 20:
   print('20')
else:
   print('other')
```

[실행 결과]

```
20
```

① either
② elif
③ else if
④ else

[해설]
Python에서 if문에 조건을 추가할 때 사용하는 예약어는 elif입니다. 사용된 코드의 의미는 다음과 같습니다.

```
❶ x = 20
❷ if x == 10:
❸    print('10')
❹ elif x == 20:
❺    print('20')
❻ else:
❼    print('other')
```

❶ 변수 x에 20을 저장합니다.
❷ x가 10이면 ❸번으로 이동하고, 아니면 ❹번으로 이동한다. x의 값은 10이 아니므로 ❹번으로 이동합니다.
❹ x가 20이면 ❺번으로 이동하고, 아니면 ❻번의 다음 줄인 ❼번으로 이동합니다. x의 값은 20이므로 ❺번으로 이동합니다.
❺ 화면에 20을 출력합니다.
결과 20

34. 다음 파이썬 코드에서 '53t44'를 입력했을 때 출력 결과는?

```
a, b = map(int, input( ).split("t"));
print(a, b)
```

① 53 t 44
② 53t44
③ 53 44
④ 53, 44

[해설]
사용된 코드의 의미는 다음과 같습니다.

```
❶ a, b = map(int, input( ).split("t"));
❷ print(a, b)
```

❶ input( ) 메소드로 입력받은 값을 "t"를 기준으로 분리한 후 정수로 변환하여 a, b에 저장됩니다. 문제에서 "53t44"를 입력하였으므로, 53이 a에 44가 b에 저장됩니다.
- **map( )** : 2개 이상의 값을 원하는 자료형으로 변환할 때 사용하는 함수
- **input( ).split('분리문자')**
  - 입력받은 값을 '분리문자'를 기준으로 분리하여 반환합니다.
  - '분리문자'를 생략하면 공백을 기준으로 값을 분리합니다.

❷ a와 b를 출력합니다. Python의 print( ) 메소드에서 2개 이상의 값을 출력할 때, sep 속성값을 정의하지 않으면 기본값이 공백이므로 다음과 같이 출력됩니다.

결과 53 44

35. 다음 쉘 스크립트의 의미로 옳은 것은?

```
until who | grep wow
do
sleep 5
done
```

① wow 사용자가 로그인한 경우에만 반복문을 수행한다.
② **wow 사용자가 로그인할 때까지 반복문을 수행한다.**
③ wow 문자열을 복사한다.
④ wow 사용자에 대한 정보를 무한 반복하여 출력한다.

[해설]
해당 코드는 wow 사용자가 접속할 때까지 5초마다 확인하는 스크립트입니다.

```
❶ until who | grep wow
❷ do
❸ sleep 5
❹ done
```

❶ wow 사용자가 로그인할 때까지 ❷~❹번을 반복 수행합니다.
❷ 반복문의 시작입니다. ❹번까지가 반복문입니다.
❸ 5초간 프로그램 수행을 멈춥니다.
❹ 반복문의 끝입니다. ❷번으로 돌아갑니다.

36. 다음 SQL문의 실행 결과는?

```
SELECT 가격 FROM 도서가격
WHERE 책번호 = (SELECT 책번호 FROM 도서 WHERE 책명='자료구조');
```

[도서]

책번호	책명
111	운영체제
222	자료구조
333	컴퓨터구조

[도서가격]

책번호	가격
111	20,000
222	25,000
333	10,000
444	15,000

① 10,000
② 15,000
③ 20,000
④ 25,000

[해설]
문제의 질의문은 하위 질의가 있는 질의문입니다. 먼저 WHERE 조건에 지정된 하위 질의의 SELECT 문을 검색합니다. 그리고 검색 결과를 본 질의의 조건에 있는 '책번호' 속성과 비교합니다.

❶ **SELECT 책번호 FROM 도서 WHERE 책명 = '자료구조';** : <도서> 테이블에서 '책명' 속성의 값이 "자료구조"와 같은 레코드의 '책번호' 속성의 값을 검색합니다. 결과는 222입니다.

❷ **SELECT 가격 FROM 도서가격 WHERE 책번호 = 222;** : <도서가격> 테이블에서 '책번호' 속성의 값이 222와 같은 레코드의 '가격' 속성의 값을 검색합니다. 결과는 25,000입니다.

37. 학적 테이블에서 전화번호가 Null 값이 아닌 학생명을 모두 검색할 때, SQL 구문으로 옳은 것은?

① SELECT 학생명 FROM 학적 WHERE 전화번호 DON'T NULL;
② SELECT 학생명 FROM 학적 WHERE 전화번호 != NOT NULL;
③ **SELECT 학생명 FROM 학적 WHERE 전화번호 IS NOT NULL;**
④ SELECT 학생명 FROM 학적 WHERE 전화번호 IS NULL;

[해설]
SQL 문장은 절별로 분리하여 이해하면 쉽습니다.

SELECT 학생명	'학생명'을 표시합니다.
FROM 학적	<학적> 테이블을 대상으로 검색합니다.
WHERE 전화번호 IS NOT NULL;	'전화번호'가 NULL이 아닌 튜플만을 대상으로 합니다.

※ NULL 값을 질의할 때는 IS NULL, NULL 값이 아닐 경우는 IS NOT NULL을 사용합니다.

38. 테이블 R1, R2에 대하여 다음 SQL문의 결과는?

```
(SELECT 학번 FROM R1)
INTERSECT
(SELECT 학번 FROM R2)
```

[R1] 테이블

학번	학점 수
20201111	15
20202222	20

[R2] 테이블

학번	과목번호
20202222	CS200
20203333	CS300

①

학번	학점 수	과목번호
20202222	20	CS200

②

학번
20202222

③

학번
20201111
20202222
20203333

④

학번	학점 수	과목번호
20201111	15	NULL
20202222	20	CS200
20203333	NULL	CS300

[해설]

• INTERSECT는 두 SELECT문의 조회 결과 중 공통된 행만 출력하는 집합 연산자입니다.

• SELECT 학번 FROM R1과 SELECT 학번 FROM R2의 결과는

학번
20201111
20202222

와

학번
20202222
20203333

이므로, 공통된 행인

학번
20202222

가 결과로 출력됩니다.

39. 다음 R1과 R2의 테이블에서 아래의 실행 결과를 얻기 위한 SQL문은?

[R1] 테이블

학번	이름	학년	학과	주소
1000	홍길동	1	컴퓨터공학	서울
2000	김철수	1	전기공학	경기
3000	강남길	2	전자공학	경기
4000	오말자	2	컴퓨터공학	경기
5000	장미화	3	전자공학	서울

[R2] 테이블

학번	과목번호	과목이름	학점	점수
1000	C100	컴퓨터구조	A	91
2000	C200	데이터베이스	A+	99
3000	C100	컴퓨터구조	B+	89
3000	C200	데이터베이스	B	85
4000	C200	데이터베이스	A	93
4000	C300	운영체제	B+	88
5000	C300	운영체제	B	82

[실행 결과]

과목번호	과목이름
C100	컴퓨터구조
C200	데이터베이스

① SELECT 과목번호, 과목이름 FROM R1, R2 WHERE R1.학번 = R2.학번 AND R1.학과 = '전자공학' AND R1.이름 = '강남길';

② SELECT 과목번호, 과목이름 FROM R1, R2 WHERE R1.학번 = R2.학번 OR R1.학과 = '전자공학' OR R1.이름 = '홍길동';

③ SELECT 과목번호, 과목이름 FROM R1, R2 WHERE R1.학번 = R2.학번 AND R1.학과 = '컴퓨터공학' AND R1.이름 = '강남길';

④ SELECT 과목번호, 과목이름 FROM R1, R2 WHERE R1.학번 = R2.학번 OR R1.학과 = '컴퓨터공학' OR R1.이름 = '홍길동';

[해설]

질의문의 각 절을 분리하여 이해하세요.

```
❶ SELECT 과목번호, 과목이름
❷ FROM R1, R2
❸ WHERE R1.학번 = R2.학번
❹   AND R1.학과 = '전자공학'
❺   AND R1.이름 = '강남길';
```

❶ '과목번호'와 '과목이름'을 표시합니다.

❷ <R1>, <R2> 테이블을 대상으로 검색합니다.

❸ <R1> 테이블의 '학번'이 <R2> 테이블의 '학번'과 같고,

❹ <R1> 테이블의 '학과'가 "전자공학"이며,

❺ <R1> 테이블의 '이름'이 "강남길"인 튜플만을 대상으로 합니다.

40. 다음 [조건]에 부합하는 SQL문을 작성하고자 할 때, [SQL문]의 빈칸에 들어갈 내용으로 옳은 것은? (단, '팀코드' 및 '이름'은 속성이며, '직원'은 테이블이다.)

[조건]

이름이 '정도일'인 팀원이 소속된 팀코드를 이용하여 해당 팀에 소속된 팀원들의 이름을 출력하는 SQL 문 작성

[SQL문]

```
SELECT 이름
FROM 직원
WHERE 팀코드 = (                    ) ;
```

① WHERE 이름 = '정도일'
② SELECT 팀코드 FROM 이름 WHERE 직원 = '정도일'
③ WHERE 직원 = '정도일'
④ SELECT 팀코드 FROM 직원 WHERE 이름 = '정도일'

[해설]
문제의 질의문은 하위 질의가 있는 질의문입니다. 먼저 WHERE 조건에 지정된 하위 질의의 SELECT 문을 검색합니다. 그리고 검색 결과를 본 질의의 조건에 있는 '팀코드' 속성과 비교합니다.

❶ **SELECT 팀코드 FROM 직원 WHERE 이름 = '정도일'** : <직원> 테이블에서 '이름' 속성의 값이 "정도일"과 같은 레코드의 '팀코드' 속성의 값을 검색합니다.

❷ **SELECT 이름 FROM 직원 WHERE 팀코드 = ❶;** : <직원> 테이블에서 '팀코드' 속성의 값이 ❶의 결과와 같은 레코드의 '이름' 속성의 값을 검색합니다.

41. [player] 테이블에는 player_name, team_id, height 컬럼이 존재한다. 아래 SQL문에서 문법적 오류가 있는 부분은?

```
(1) SELECT player_name, height
(2) FROM player
(3) WHERE team_id = 'korea'
(4) AND height BETWEEN 170 OR 180;
```

① (1)
② (2)
③ (3)
④ (4)

[해설]
BETWEEN은 '**BETWEEN** A **AND** B'의 형식으로 사용됩니다.

```
❶ SELECT player_name, height
❷ FROM player
❸ WHERE team_id = 'korea'
❹ AND height BETWEEN 170 AND 180;
```

❶ 'player_name', 'height'를 표시합니다.
❷ <player> 테이블을 대상으로 검색합니다.
❸ 'team_id'가 "korea"이고
❹ 'height'의 값이 170~180 사이인 자료만을 대상으로 합니다.

42. 관계 데이터베이스인 테이블 R1에 대한 아래 SQL 문의 실행 결과로 옳은 것은?

[R1]

학번	이름	학년	학과	주소
1000	홍길동	1	컴퓨터공학	서울
2000	김철수	1	전기공학	경기
3000	강남길	2	전기공학	경기
4000	오말자	2	컴퓨터공학	경기
5000	장미화	3	전기공학	서울

[SQL 문]

```
SELECT DISTINCT 학년 FROM R1;
```

①

학년
1
1
2
2
3

②

학년
1
2
3

③

이름	학년
홍길동	1
김철수	1
강남길	2
오말자	2
장미화	3

④

이름	학년
홍길동	1
강남길	2
장미화	3

[해설]

- **SELECT DISTINCT 학년** : '학년'을 검색하되, 같은 '학년' 속성의 값은 한 번만 표시합니다.
- **FROM R1;** : <R1> 테이블을 대상으로 검색합니다.

43. 결과 값이 아래와 같을 때 SQL 질의로 옳은 것은?

[공급자] Table

공급자번호	공급자명	위치
16	대신공업사	수원
27	삼진사	서울
39	삼양사	인천
62	진아공업사	대전
70	신촌상사	서울

[결과]

공급자번호	공급자명	위치
16	대신공업사	수원
70	신촌상사	서울

① SELECT * FROM 공급자 WHERE 공급자명 LIKE '%신%';

② SELECT * FROM 공급자 WHERE 공급자명 LIKE '대%';

③ SELECT * FROM 공급자 WHERE 공급자명 LIKE '%사';

④ SELECT * FROM 공급자 WHERE 공급자명 IS NOT NULL;

[해설]

① LIKE '%신%' : 공급자명에 "신"이 포함된 레코드

공급자번호	공급자명	위치
16	대신공업사	수원
70	신촌상사	서울

② LIKE '대%' : 공급자명이 "대"로 시작하는 레코드

공급자번호	공급자명	위치
16	대신공업사	수원

③ LIKE '%사' : 공급자명이 "사"로 끝나는 레코드

공급자번호	공급자명	위치
16	대신공업사	수원
27	삼진사	서울
39	삼양사	인천
62	진아공업사	대전
70	신촌상사	서울

④ IS NOT NULL : 공급자명이 NULL이 아닌 레코드

공급자번호	공급자명	위치
16	대신공업사	수원
27	삼진사	서울
39	삼양사	인천
62	진아공업사	대전
70	신촌상사	서울

44. 다음 SQL문의 실행 결과는?

[R1 테이블]

학번	이름	학년	학과	주소
1000	홍길동	4	컴퓨터	서울
2000	김철수	3	전기	경기
3000	강남길	1	컴퓨터	경기
4000	오말자	4	컴퓨터	경기
5000	장미화	2	전자	서울

[R2 테이블]

학번	과목번호	학점	점수
1000	C100	A	91
1000	C200	A	94
2000	C300	B	85
3000	C400	A	90
3000	C500	C	75
3000	C100	A	90
4000	C400	A	95
4000	C500	A	91
4000	C100	B	80
4000	C200	C	74
5000	C400	B	85

[SQL 문]

```
SELECT 이름
FROM R1
WHERE 학번 IN
    (SELECT 학번
     FROM R2
     WHERE 과목번호 = 'C100' );
```

①

이름
홍길동
강남길
장미화

②

이름
홍길동
강남길
오말자

③

이름
홍길동
김철수
강남길
오말자
장미화

④

이름
홍길동
김철수

[해설]

<R2> 테이블에서 '과목번호' 속성이 "C100"인 학번을 <R1> 테이블에서 찾아 '이름' 속성을 출력합니다.

```
❷ SELECT 이름 FROM R1 WHERE 학번 IN
❶ (SELECT 학번 FROM R2 WHERE 과목번호 = 'C100' );
```

❶ **SELECT 학번 FROM R2 WHERE 과목번호 = 'C100'** : <R2> 테이블에서 '과목번호'가 "C100"인 튜플의 '학번'을 검색합니다. 결과는 1000, 3000, 4000입니다.

❷ **SELECT 이름 FROM R1 WHERE 학번 IN (❶)** : <R1> 테이블에서 '학번'이 1000, 3000, 4000인 튜플의 '이름'을 검색합니다. 결과는 "홍길동", "강남길", "오말자"입니다.

45. 다음 SQL문의 실행 결과로 생성되는 튜플 수는?

```
SELECT 급여 FROM 사원;
```

<사원> 테이블

사원ID	사원명	급여	부서ID
101	박철수	30000	1
102	한나라	35000	2
103	김감동	40000	3
104	이구수	35000	2
105	최초록	40000	3

① 1

② 3

③ 4

④ 5

[해설]

• **SELECT 급여** : '급여' 필드를 표시합니다.

• **FROM 사원** : <사원> 테이블의 자료를 검색합니다.

∴ WHERE문이 없으므로 <사원> 테이블에서 '급여' 필드의 전체 레코드를 검색합니다.

<실행 결과>

급여
30000
35000
40000
35000
40000

46. 다음 SQL문의 실행 결과는?

```
SELECT 과목이름
FROM 성적
WHERE EXISTS (
      SELECT 학번
      FROM 학생
      WHERE 학생.학번 = 성적.학번
         AND 학생.학과 IN ('전산', '전기')
         AND 학생.주소 = '경기');
```

[학생] 테이블

학번	이름	학년	학과	주소
1000	김철수	1	전산	서울
2000	고영준	1	전기	경기
3000	유진호	2	전자	경기
4000	김영진	2	전산	경기
5000	정현영	3	전자	서울

[성적] 테이블

학번	과목번호	과목이름	학점	점수
1000	A100	자료구조	A	91
2000	A200	DB	A+	99
3000	A100	자료구조	B+	88
3000	A200	DB	B	85
4000	A200	DB	A	94
4000	A300	운영체제	B+	89
5000	A300	운영체제	B	88

①

과목이름
DB

②

과목이름
DB
DB

③

과목이름
DB
DB
운영체제

④

과목이름
DB
운영체제

[해설]

<학생> 테이블에서 '학과'가 "전산" 또는 "전기"이고, '주소'가 "경기"인 튜플의 '학번'을 <성적> 테이블에서 조회한 후 해당 튜플의 '과목이름'을 출력하면 됩니다. 결과로 '학번'이 2000과 4000인 사람의 '과목이름'이 차례대로 출력됩니다.

```
❶ SELECT 과목이름
❷ FROM 성적
❸ WHERE EXISTS (
❹       SELECT 학번
❺       FROM 학생
❻       WHERE 학생.학번  =  성적.학번
❼          AND 학생.학과 IN ('전산', '전기')
❽          AND 학생.주소 = '경기');
```

❶ '과목이름'을 표시합니다.
❷ <성적> 테이블에서 검색합니다.
❸ 하위 질의에 결과가 한 건이라도 있으면 참(True), 없으면 거짓(False을) 반환합니다.
❹ '학번'을 표시합니다.
❺ <학생> 테이블에서 검색합니다.
❻ <학생> 테이블의 '학번'과 <성적> 테이블의 '학번'이 같고,
❼ <학생> 테이블의 '학과'가 "전산" 또는 "전기"이며,
❽ <학생> 테이블의 '주소'가 "경기"인 튜플만을 대상으로 합니다.

47. 다음 테이블을 보고 강남지점의 판매량이 많은 제품부터 출력되도록 할 때 다음 중 가장 적절한 SQL 구문은? (단, 출력은 제품명과 판매량이 출력되도록 한다.)

<푸드> 테이블

지점명	제품명	판매량
강남지점	비빔밥	500
강북지점	도시락	300
강남지점	도시락	200
강남지점	미역국	550
수원지점	비빔밥	600
인천지점	비빔밥	800
강남지점	잡채밥	250

① SELECT 제품명, 판매량 FROM 푸드 ORDER BY 판매량 ASC;
② SELECT 제품명, 판매량 FROM 푸드 ORDER BY 판매량 DESC;
③ SELECT 제품명, 판매량 FROM 푸드 WHERE 지점명 = '강남지점' ORDER BY 판매량 ASC;
④ SELECT 제품명, 판매량 FROM 푸드 WHERE 지점명 = '강남지점' ORDER BY 판매량 DESC;

[해설]

SELECT 제품명, 판매량	'제품명'과 '판매량'을 표시합니다.
FROM 푸드	<푸드> 테이블을 대상으로 검색합니다.
WHERE 지점명 = '강남지점'	'지점명'이 "강남지점"인 튜플만을 대상으로 합니다.
ORDER BY 판매량 DESC;	'판매량'을 기준으로 내림차순으로 정렬합니다.

48. 테이블 R과 S에 대한 SQL문이 실행되었을 때, 실행 결과로 옳은 것은?

[R]

A	B
1	A
3	B

[S]

A	B
1	A
2	B

```
SELECT A FROM R
UNION ALL
SELECT A FROM S;
```

① 1

② 3
2

③ 1
3

④ 1
3
1
2

[해설]
문제에 제시된 질의문은 집합 연산자 UNION ALL을 이용한 통합 질의로, 여러 테이블의 필드 값을 통합하여 표시하되 중복된 레코드도 그대로 표시합니다.

49. 다음 SQL문에서 빈 칸에 들어갈 내용으로 옳은 것은?

```
UPDATE 회원 (      ) 전화번호 = '010-14'
WHERE 회원번호 = 'N4';
```

① FROM
② SET
③ INTO
④ TO

[해설]
UPDATE문은 항상 SET과 함께 사용됩니다.

UPDATE 회원	<회원> 테이블을 갱신합니다.
SET 전화번호 = '010-14'	'전화번호'를 "010-14"로 갱신합니다.
WHERE 회원번호 = 'N4';	'회원번호'가 "N4"인 튜플만을 대상으로 합니다.

50. 다음 SQL문에서 사용된 BETWEEN 연산의 의미와 동일한 것은?

```
SELECT *
FROM 성적
WHERE (점수 BETWEEN 90 AND 95)
    AND 학과 = '컴퓨터공학과';
```

① 점수 >= 90 AND 점수 <= 95
② 점수 > 90 AND 점수 < 95
③ 점수 > 90 AND 점수 <= 95
④ 점수 >= 90 AND 점수 < 95

[해설]
- **SELECT *** : 모든 필드를 표시합니다.
- **FROM 성적** : <성적> 테이블의 자료를 검색합니다.
- **WHERE (점수 BETWEEN 90 AND 95)** : 점수가 90~95 사이이고
- **AND 학과 = '컴퓨터공학과';** : '학과'가 "컴퓨터공학과"인 자료만을 대상으로 합니다.

∴ <성적> 테이블에서 점수가 90~95 사이이고 '학과'가 '컴퓨터공학과'인 모든 필드를 검색합니다.

51. 사용자 'PARK'에게 테이블을 생성할 수 있는 권한을 부여하기 위한 SQL문의 구성으로 빈칸에 적합한 내용은?

```
GRANT [                ] PARK;
```

① CREATE TABLE TO
② CREATE TO
③ CREATE FROM
④ CREATE TABLE FROM

[해설]

GRANT CREATE	생성(CREATE) 권한을 부여합니다.
TABLE	테이블을 생성할 수 있는 권한을 부여합니다.
TO PARK	ID가 'PARK'인 사용자에게 부여합니다.

52. STUDENT 테이블에 독일어과 학생 50명, 중국어과 학생 30명, 영어영문학과 학생 50명의 정보가 저장되어 있을 때, 다음 두 SQL문의 실행 결과 튜플 수는? (단, DEPT 컬럼은 학과명)

ⓐ SELECT DEPT FROM STUDENT;
ⓑ SELECT DISTINCT DEPT FROM STUDENT;

① ⓐ 3, ⓑ 3
② ⓐ 50, ⓑ 3
③ ⓐ 130, ⓑ 3
④ ⓐ 130, ⓑ 130

[해설]
ⓐ STUDENT 테이블에서 DEPT를 검색합니다. 총 130개의 튜플이 들어 있고 검색 조건이 없으므로 튜플의 수는 130개입니다.
ⓑ STUDENT 테이블에서 DEPT를 검색하는 데 중복된 결과는 처음의 한 개만 검색에 포함시킵니다. 독일어과 50개 튜플의 DEPT 속성의 값이 같으므로 1개, 중국어과 30개 튜플의 DEPT 속성의 값이 같으므로 1개, 영어영문학과 50개 튜플의 DEPT 속성의 값이 같으므로 1개를 검색에 포함시키므로 총 3개의 튜플이 검색됩니다.

53. DBA가 사용자 PARK에게 [STUDENT] 테이블의 데이터를 갱신할 수 있는 시스템 권한을 부여하는 SQL문을 작성하려고 한다. 다음에 주어진 SQL문의 빈칸을 알맞게 채운 것은?

SQL > GRANT ㉠ ㉡ STUDENT TO PARK;

① ㉠ INSERT, ㉡ INTO
② ㉠ ALTER, ㉡ TO
③ ㉠ UPDATE, ㉡ ON
④ ㉠ REPLACE, ㉡ IN

[해설]
GRANT문의 기본 형식은 'GRANT 권한_리스트 ON 개체 TO 사용자 [WITH GRANT OPTION];'이지만, 부여받을 권한을 다른 사용자에게 다시 부여할 수 있는 권한에 대한 언급이 없으므로 '[WITH GRANT OPTION]'을 생략하고 작성하면 됩니다.

GRANT UPDATE	갱신(UPDATE) 권한을 부여합니다.
ON STUDENT	<STUDENT> 테이블에 대한 권한을 부여합니다.
TO PARK	'PARK'라는 사용자에게 부여합니다.

54. 테이블 두 개를 조인하여 뷰 V_1을 정의하고, V_1을 이용하여 뷰 V_2를 정의하였다. 다음 명령 수행 후 결과로 옳은 것은?

```
DROP VIEW V_1 CASCADE;
```

① V_1만 삭제된다.
② V_2만 삭제된다.
③ V_1과 V_2 모두 삭제된다.
④ V_1과 V_2 모두 삭제되지 않는다.

[해설]
CASCADE는 제거할 요소를 참조하는 다른 모든 개체를 함께 제거하므로 V_1을 제거하면 V_2도 함께 삭제됩니다.

55. 다음은 스택의 자료 삭제 알고리즘이다. ⓐ에 들어갈 내용으로 옳은 것은? (단, Top : 스택 포인터, S : 스택의 이름)

```
if Top = 0 Then
    ( ⓐ )
Else {
    remove S(Top)
    Top = Top - 1
}
```

① Overflow
② Top = Top + 1
③ Underflow
④ Top = Top

[해설]
스택에서 자료의 삭제가 발생했을 때 자료의 가장 바깥쪽을 가리키는 스택 포인터가 0이면 언더플로(Underflow)가 발생하고, 아니면 현재 스택 포인터의 위치에 있는 자료가 삭제되면서 스택 포인터의 값이 1 감소합니다.

56. 관계 대수식을 SQL 질의로 옳게 표현한 것은?

$\pi_{이름}(\sigma_{학과 = '교육'}(학생))$

① SELECT 학생 FROM 이름 WHERE 학과 = '교육';
② SELECT 이름 FROM 학생 WHERE 학과 = '교육';
③ SELECT 교육 FROM 학과 WHERE 이름 = '학생';
④ SELECT 학과 FROM 학생 WHERE 이름 = '교육';

[해설]
- $\pi_{이름}$: '이름' 필드를 표시합니다.
- $\sigma_{학과 = '교육'}$: '학과'가 "교육"인 자료만을 대상으로 합니다.
- **(학생)** : <학생> 테이블의 자료를 검색합니다.

∴ 교육과 학생의 '이름'을 검색합니다.

57. 사용자 X1에게 department 테이블에 대한 검색 연산을 회수하는 명령은?

① delete select on department to X1;

② remove select on department from X1;

③ revoke select on department from X1;

④ grant select on department from X1;

[해설]

사용자로부터 권한을 취소(회수)하는 명령어는 revoke입니다.

revoke select	검색(select) 권한을 취소합니다.
on department	<department> 테이블에 대한 권한을 취소합니다.
from X1;	사용자 'X1'에 대한 권한을 취소합니다.